Projeto 1: Parte A - "Importação e limpeza de dados no Python

---

Texto por:

Lucas Andrei Campos-Siva

Cientista de Dados / Business Intelligence / Analista de Dados

LinkedIn: https://www.linkedin.com/in/lucas-andrei-campos-silva/

E-mail: andrei.10@hotmail.com

Portifólio de projetos em Data Science: https://github.com/Campos-Silva

---

Objetivos desse projeto:

Nesse projeto irei demonstrar como são realizadas 2 etapas essenciais de qualquer trabalho de Data Science, que são:

- (1) Importação e conhecimento prévio sobre um conjunto de dados;
- (2) Limpeza dos dados / Pré-processamento.

Detalhamento do Dataset estudado

- Para isso irei trabalhar com um Dataset que vem do site: https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho
- O dataset possui dados de carros usados o qual permite fazer diversas análises em Data Science, inclusive criar modelos preditivos de machine learning.

Perguntas centrais do Dataset:

Nesse dataset podemos explorar a seguintes pergunta central:

- O valor de preço de venda final desses carros usados está relacionada aos seus atributos?
- Se sim, quais os atributos que irão influenciar nesse valor de preço?

- Para respondê-las primeiro tenho de realizar a

"(1) Importação e conhecimento prévio sobre um conjunto de dados" e "(2) Limpeza dos dados / Pré-processamento".

- Só após concluídas essas etapas que posso dar prosseguimento a etapas subsequentes de Data Science, como "Análises Exploratórias do dataset" e "Criação de Modelos de Machine Learning".

- Essas outras etapas serão realizadas em um próximo projeto.

Bibliotecas utilizadas nesse projeto:

- pandas
- numpy
- missingno
- matplotlib

(1) Importação e conhecimento prévio sobre um conjunto de dados

In [ ]:
```
#Vou trabalhar com a biblioteca pandas para importar e explorar inicialmente o Datase

#Importando a biblioteca pandas

import pandas as pd
```

In [ ]:
```
#Importando a biblioteca Numpy

import numpy as np
```

In [ ]:
```
# Importando via github

import pandas as pd
url = 'https://raw.githubusercontent.com/Campos-Silva/Projeto_01_Parte_A_Importacao-e
carros = pd.read_csv(url)
carros.head()
```

Out[ ]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	10
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	

Aqui irei começar a entender como está configurado o meu conjunto de dados.

In [ ]:
```
#Vou identificar quantas linhas e colunas existem nesse conjunto de dados

carros.shape
```

Out[ ]:
```
(8128, 13)
```

In [ ]:
```
#Quero visualizar as 5 primeiras linhas para identificar quais são as variáveis que e

carros.head(5)
```

Out[ ]:

	name	year	selling_price	km_driven	fuel	seller_type	transmission	owner	mileage	engine	max
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Individual	Manual	First Owner	23.4 kmpl	1248 CC	
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Individual	Manual	Second Owner	21.14 kmpl	1498 CC	10
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Individual	Manual	Third Owner	17.7 kmpl	1497 CC	
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Individual	Manual	First Owner	23.0 kmpl	1396 CC	
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Individual	Manual	First Owner	16.1 kmpl	1298 CC	

In [ ]:
```
#Vou alterar o nome das colunas para nomes em português

carros.columns = "nome_completo_do_carro", "ano_de_venda", "preco_de_venda" , "kilome
carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda	kilometragem	tipo_combustivel	tipo_do_vend
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Indiv
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Indiv
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Indiv
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Indiv
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Indiv

Vou identificar quais são os tipos de dados para cada variável.

In [ ]:
```
carros.dtypes
```

Out[ ]:
```
nome_completo_do_carro     object
ano_de_venda                int64
preco_de_venda              int64
kilometragem                int64
tipo_combustivel           object
tipo_do_vendedor           object
```

```
transmissao                object
tipo_do_dono               object
consumo_do_combustivel     object
motor                      object
potencia_do_motor          object
torque                     object
assentos                  float64
dtype: object
```

No site https://pbpython.com/pandas_dtypes.html, encontramos uma imagem que ilustra o significado para os diferentes tipos de valores da biblioteca pandas, por meio da função "dtype".

Pandas dtype	Python type	NumPy type	Usage
object	str or mixed	string_, unicode_, mixed types	Text or mixed numeric and non-numeric values
int64	int	int_, int8, int16, int32, int64, uint8, uint16, uint32, uint64	Integer numbers
float64	float	float_, float16, float32, float64	Floating point numbers
bool	bool	bool_	True/False values
datetime64	datetime	datetime64[ns]	Date and time values
timedelta[ns]	NA	NA	Differences between two datetimes
category	NA	NA	Finite list of text values

Por meio dessa referência e da função realizada identifiquei que há os seguintes tipos de valores no dataset:

- "object", o qual é dado é dado para texto e valores não numéricos;
- "int64", que se refere a números inteiros;
- "float64", que se refere a valores com decimais.

Assim, esse dataset possui:

- **4** variáveis numéricas que são: "ano", "preco_de_venda", "kilometragem" e "assentos";
- **9** variáveis categóricas que são: "nome_completo_do_carro", "tipo_combustivel", "tipo_do_vendedor", "transmissao", "dono", "consumo_do_combustivel", "motor", "potencia_do_motor" e "torque".

Resumo de estatísticas descritivas para o dataset

- Informações para valores numéricos

In [ ]:
```
carros.describe()
```

Out[ ]:

	ano_de_venda	preco_de_venda	kilometragem	assentos
count	8128.000000	8.128000e+03	8.128000e+03	7907.000000
mean	2013.804011	6.382718e+05	6.981951e+04	5.416719
std	4.044249	8.062534e+05	5.655055e+04	0.959588
min	1983.000000	2.999900e+04	1.000000e+00	2.000000
25%	2011.000000	2.549990e+05	3.500000e+04	5.000000
50%	2015.000000	4.500000e+05	6.000000e+04	5.000000
75%	2017.000000	6.750000e+05	9.800000e+04	5.000000

max	2020.000000	1.000000e+07	2.360457e+06	14.000000

Resumo de estatísticas descritivas para o dataset

- Incluindo todas as variáveis, inclusive as categóricas

In [ ]:
```
carros.describe(include="all")
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda	kilometragem	tipo_combustivel	tipo_do
count	8128	8128.000000	8.128000e+03	8.128000e+03	8128	
unique	2058	NaN	NaN	NaN	4	
top	Maruti Swift Dzire VDI	NaN	NaN	NaN	Diesel	
freq	129	NaN	NaN	NaN	4402	
mean	NaN	2013.804011	6.382718e+05	6.981951e+04	NaN	
std	NaN	4.044249	8.062534e+05	5.655055e+04	NaN	
min	NaN	1983.000000	2.999900e+04	1.000000e+00	NaN	
25%	NaN	2011.000000	2.549990e+05	3.500000e+04	NaN	
50%	NaN	2015.000000	4.500000e+05	6.000000e+04	NaN	
75%	NaN	2017.000000	6.750000e+05	9.800000e+04	NaN	
max	NaN	2020.000000	1.000000e+07	2.360457e+06	NaN	

(2) Limpeza dos dados / Pré-processamento.

Essa etapa de limpeza dos dados é uma das mais importantes etapas em trabalhos de Data Science.

Isso porque é nessa etapa que:

- (1) identificamos e lidamos com a existência de possíveis valores ausentes que possam prejudicar as análises posteriores;
- (2) identificamos se os dados de cada variável estão formatados corretamente. Dados devidamente formatados garantem que as análises preditivas de machine learning ajudem a responder as perguntas centrais de forma eficaz;
- (3) identificamos se os dados estão normalizados ou não. A normalização é uma etapa importante, uma vez que torna valores de variáveis específicas dentro de uma mesma escala de proporções. Isso permite que modelos preditivos possam ser criados de forma eficaz;
- (4) compartimentalizamos dados, ato também conhecido como "Binning". Binning é o ato de compartimentalizar dados de uma variável quantitativa em determinados grupos através de seus valores. Tal conversão ajuda a melhor entender a distribuição desses valores.
- (5) transformamos variáveis categóricas para variáveis numéricas, criando variáveis indicadoras. É com o uso de variáveis indicadoras que conseguimos utilizar variáveis categóricas para análises de regressão.

Nesse tópico irei realizar as duas primeiras tarefas descritas acima:

(1) Identificar e manejar valores ausentes

Há várias formas para lidar com dados ausentes. E pode ser feita, por exemplo, de 4 formas como descritas a seguir:

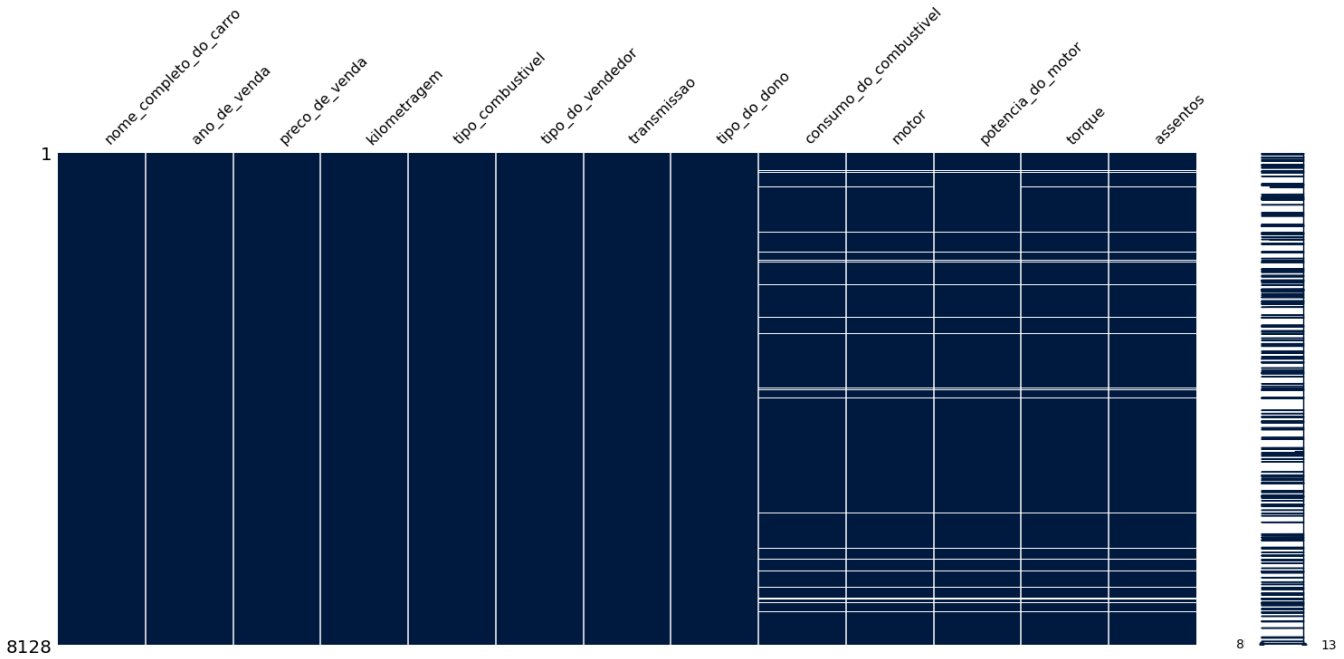
- A - Rever os dados originais e checar se na hora de passar para a planilha um determinado valor não foi "perdido";
- B - Substituir o valor ausente por outro valor;
- C - Remover as linhas ou apenas as colunas que contêm esses valores ausentes.
- D - Deixar da forma que está;

Temos de checar inicialmente se há valores ausentes nesses dados

In [ ]:
```
#Identificando graficamente a ausência de valores no dataset

import missingno as msg
msg.matrix(carros, color = (0, 0.1, 0.25))
```

Out[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0e7ef97e50>
```



In [ ]:
```
# Identificando quantos valores ausentes existem para cada variável
carros.isnull().sum()
```

Out[ ]:
```
nome_completo_do_carro      0
ano_de_venda                0
preco_de_venda              0
kilometragem                0
tipo_combustivel            0
tipo_do_vendedor            0
transmissao                 0
tipo_do_dono                0
consumo_do_combustivel    221
motor                     221
potencia_do_motor         215
torque                    222
assentos                  221
dtype: int64
```

Através do resultado da função acima, identifiquei que o conjunto de dados têm 5 variáveis com valores nulos, que são: (1) "consumo_do_combustivel ", (2) "motor", (3) "potencia_do_motor ", (4) "torque" e (5) "assentos".

Como esses dados foram coletados por outra pessoa, eu não tenho acesso ao banco original dos dados. Assim, não tenho como checar se algum dado foi "perdido" no momento que foi passado para a planilha atual.

Para esse projeto vou demonstrar como posso lidar usando as seguintes formas:

- (A) - Substituir o valor ausente por outro;
- (B) - Remover a linha / ou coluna" em que está esse valor ausente.

Para exemplificar como lidamos com dados ausentes vou utilizar apenas a variável numérica "assentos".

(A) - Substituir o valor ausente por outro

Para substituir um valor ausente por algum outro podemos, por exemplo, substituir pela média de todos esses valores da variável. É isso que irei fazer.

In [ ]:
```
#Identificando a média da variável "assentos"

mean = carros["assentos"].mean()

mean
```

Out[ ]:
```
5.41671936259011
```

In [ ]:
```
#Substituindo os valores ausentes dessa variável por sua média

carros["assentos"].fillna(value=mean, inplace=True)
```

In [ ]:
```
# Identificando quantos valores ausentes existem para cada variável, incluindo a vari
carros.isnull().sum()
```

Out[ ]:
```
nome_completo_do_carro      0
ano_de_venda                0
preco_de_venda              0
kilometragem                0
tipo_combustivel            0
tipo_do_vendedor            0
transmissao                 0
tipo_do_dono                0
consumo_do_combustivel    221
motor                     221
potencia_do_motor         215
torque                    222
assentos                    0
dtype: int64
```

Por meio das funções acima é possível identificar que não há mais nenhum valor ausente para a variável "assentos".

(B) - Remover a linha / ou coluna" em que está esse valor ausente

Importando novamente os dados iniciais

In [ ]:
```
# Importando via github
```

```
import pandas as pd
url = 'https://raw.githubusercontent.com/Campos-Silva/Projeto_01_Parte_A_Importacao-e
carros = pd.read_csv(url)
carros.head()

#Vou alterar o nome das colunas para nomes em português

carros.columns = "nome_completo_do_carro", "ano_de_venda","preco_de_venda" , "kilomet
carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda	kilometragem	tipo_combustivel	tipo_do_vend
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Indiv
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Indiv
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Indiv
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Indiv
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Indiv

In [ ]:
```
#Removendo apenas as linhas com valores ausentes da variável "assentos"

carros.dropna(subset=["assentos"], axis=0, inplace = True)

carros = carros.dropna(subset=["assentos"], axis=0,)
```

In [ ]:
```
#Checando novamente para ver se há valores ausentes nessa variável:

carros.isnull().sum()
```

Out[ ]:
```
nome_completo_do_carro    0
ano_de_venda              0
preco_de_venda            0
kilometragem              0
tipo_combustivel          0
tipo_do_vendedor          0
transmissao               0
dono                      0
consumo_do_combustivel    0
motor                     0
potencia_do_motor         0
torque                    1
assentos                  0
dtype: int64
```

- Ainda há uma variável com valor ausente. Assim, irei remover também essa linha da variável "torque".

In [ ]:
```
#Removendo apenas as linhas com valores ausentes

carros.dropna(subset=["torque"], axis=0, inplace = True)

carros = carros.dropna(subset=["torque"], axis=0,)
```

In [ ]:

```
#Checando novamente para ver se há valores ausentes nessa variável:

carros.isnull().sum()
```

Out[ ]:
```
nome_completo_do_carro    0
ano_de_venda              0
preco_de_venda            0
kilometragem              0
tipo_combustivel          0
tipo_do_vendedor          0
transmissao               0
dono                      0
consumo_do_combustivel    0
motor                     0
potencia_do_motor         0
torque                    0
assentos                  0
dtype: int64
```

Não há mais nenhuma variável com valor ausente. Assim, podemos prosseguir.

Observação:

Para as próximas etapas irei *manter essa configuração atual*, no qual as linhas com valores ausentes foram removidas.

(2) Formatação dos dados

In [ ]:
```
#Revendo novamente os dados

carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda	kilometragem	tipo_combustivel	tipo_do_vend
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Indiv
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Indiv
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Indiv
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Indiv
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Indiv

Observação:

- Irei checar todas as 13 variáveis e irei formatá-las, uma a uma caso caso eu identifique a necessidade;
- Essa formatação permitirá que as perguntas centrais mencionadas no começo do projeto sejam atingidas de forma eficaz;
- As perguntas centrais serão alvo de estudo dos próximos projetos, mas irão necessitar dos passos a seguir.

- Após formatadas eu irei salvar um novo dataframe. Isso fará com que eu tenha acesso fácil a esse novo dataset já com as devidas formatações.

Variável 1 - "nome_completo_do_carro"

Como é uma variável categórica, a melhor forma de checar se os dados estão corretamente formatados será através dos valores únicos:

In [ ]:
```
#Identificando os valores únicos dessa variável

carros.nome_completo_do_carro.unique()
```

Out[ ]:
```
array(['Maruti Swift Dzire VDI', 'Skoda Rapid 1.5 TDI Ambition',
       'Honda City 2017-2020 EXi', ..., 'Tata Nexon 1.5 Revotorq XT',
       'Ford Freestyle Titanium Plus Diesel BSIV',
       'Toyota Innova 2.5 GX (Diesel) 8 Seater BS IV'], dtype=object)
```

In [ ]:
```
carros.nome_completo_do_carro.drop_duplicates()
```

Out[ ]:
```
0                            Maruti Swift Dzire VDI
1                      Skoda Rapid 1.5 TDI Ambition
2                          Honda City 2017-2020 EXi
3                         Hyundai i20 Sportz Diesel
4                            Maruti Swift VXI BSIII
                            ...
8087                          Tata Bolt Revotron XM
8094              Tata Manza Aura (ABS) Safire BS IV
8100                      Tata Nexon 1.5 Revotorq XT
8109        Ford Freestyle Titanium Plus Diesel BSIV
8113    Toyota Innova 2.5 GX (Diesel) 8 Seater BS IV
Name: nome_completo_do_carro, Length: 1982, dtype: object
```

Essa variável "nome_completo_do_carro" está devidamente formatada.

Observação:

A variável "nome_completo_do_carro" possui uma informação importante dentro dela que é a "marca/montadora" do veículo que pode ser usada nas análises exploratórias e posteriormente na criação do modelo preditivo.

Dessa forma, eu vou colocar essa importante informação em uma nova coluna denominada "Marca". Essa etapa é conhecida como "Engenharia de Variáveis" e é um passo importante do Pré-Processamento.

In [ ]:
```
#Vou inserir a variavel Marca do veiculo a partir da variavel "nome_completo_do_carro

#Vou pegar a primeira linha

carros['nome_completo_do_carro'][0]
```

Out[ ]:
```
'Maruti Swift Dzire VDI'
```

In [ ]:
```
#Vou começar a separar valores especificos dessas strings atraves de "Engenharia de V

carros['nome_completo_do_carro'][0].split(' ')[0]
```

Out[ ]:
```
'Maruti'
```

Vou seguir a mesma linha raciocínio das linhas anteriores para criar uma função que irá separar a Marca

do carro para todas as linhas do datase e depois irei colocar essas informações novas em uma coluna chamada de "Marca".

In [ ]:
```
# tranformação de variavel: pegar apenas o titulo presente no nome

def aux(x):
    return x.split(' ')[0]

carros['Marca'] = carros['nome_completo_do_carro'].apply(aux)

carros.head()
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda	kilometragem	tipo_combustivel	tipo_do_vend
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Indiv
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Indiv
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Indiv
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Indiv
4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Indiv

Agora a Marca do Veículo foi criada

2 - Variável "ano_de_venda"

In [ ]:
```
#valores únicos:

carros.ano_de_venda.unique()
```

Out[ ]:
```
array([2014, 2006, 2010, 2007, 2017, 2001, 2011, 2013, 2005, 2009, 2016,
       2012, 2002, 2015, 2018, 2019, 2008, 2020, 1999, 2000, 2003, 2004,
       1994, 1998, 1997, 1995, 1996])
```

Variável "ano" está formatada corretamente.

3- Variável "preco_de_venda"

In [ ]:
```
#Revendo novamente os dados

carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda	kilometragem	tipo_combustivel	tipo_do_vend
0	Maruti Swift Dzire VDI	2014	450000	145500	Diesel	Indiv
1	Skoda Rapid 1.5 TDI Ambition	2014	370000	120000	Diesel	Indiv
2	Honda City 2017-2020 EXi	2006	158000	140000	Petrol	Indiv
3	Hyundai i20 Sportz Diesel	2010	225000	127000	Diesel	Indiv

4	Maruti Swift VXI BSIII	2007	130000	120000	Petrol	Indiv

Observação:

- A variável preço de venda ("preco_de_venda") é uma das mais importantes variáveis desse dataset. Isso porque é com ela que irei checar se essa variável é influenciada por outras caracteristicas desse dataset.
- Ao buscar informações sobre esse dataset, no site de referência, observei que o criador desse dataset colocou o valor de "preco_de_venda" em unidades de Rupia indiana (INR), que é a moeda corrente oficial da Índia.
- No entanto, pretendo analisar sob o ponto de vista da moeda "dólar" (aqui utilizada sob a forma contraída "USD"). Assim, terei de formatar os valores para dólares.
- Um dólar, cuja unidade é USD, equivale a aproximadamente 72,38 INR na data que escrevi esse projeto (28/05/2021). Aqui irei adotar que a proporção é de 73 INR para 1 USD.
- Dessa forma, irei dividir todos os valores da coluna "preco_de_venda" por 73 conforme função abaixo:

In [ ]:
```
#Dividindo a coluna por 73

carros["preco_de_venda"]= carros["preco_de_venda"].div(73).round(2)

#Renomeando essa coluna agora para preco_de_venda_USD

carros.rename(columns={"preco_de_venda": "preco_de_venda_USD"}, inplace=True)

carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda_USD	kilometragem	tipo_combustivel	tipo_do_
0	Maruti Swift Dzire VDI	2014	6164.38	145500	Diesel	
1	Skoda Rapid 1.5 TDI Ambition	2014	5068.49	120000	Diesel	
2	Honda City 2017-2020 EXi	2006	2164.38	140000	Petrol	
3	Hyundai i20 Sportz Diesel	2010	3082.19	127000	Diesel	
4	Maruti Swift VXI BSIII	2007	1780.82	120000	Petrol	

- Valores totais para essa variável "preco_venda_USD"

In [ ]:
```
carros.preco_de_venda_USD
```

Out[ ]:
```
0       6164.38
1       5068.49
2       2164.38
```

```
3       3082.19
4       1780.82
         ...
8123    4383.56
8124    1849.32
8125    5232.88
8126    3972.60
8127    3972.60
Name: preco_de_venda_USD, Length: 7906, dtype: float64
```

- Por meio da função acima identifiquei que essa variável agora está formatada corretamente.

Variável 4 - "kilometragem "

In [ ]:
```
#valores únicos:

carros.kilometragem.unique()
```

Out[ ]:
```
array([ 145500,  120000,  140000,  127000,   45000,  175000,    5000,
         90000,  169000,   68000,  100000,   40000,   70000,   53000,
         80000,   50000,   72000,   35000,   28000,   25000,    2388,
         16200,   10000,   15000,   42000,   60000,   76000,   28900,
         86300,   23300,   32600,   10300,   77000,   99000,   27800,
         49800,  151000,   54700,   64000,   63000,  127700,   33900,
         59000,  110000,  147000,   30000,  135000,    9850,   78000,
        170000,   49000,   32000,   38000,   44000,   12000,   55500,
         61500,  150000,   37800,  114000,   48000,   69000,   13000,
         76139,   65000,    1303,   31800,   20000,   17000,   21000,
         37000,   29500,    7500,   19000,   41000,   39000,   22000,
         47000,   72200,   49900,    9000,   11000,   95000,   18000,
         46100,   16000,    9654,   24300,   42163,    8000,   71000,
          9500,   36600,   14000,  120600,   86000,    7800,   31377,
         75000,   93000,  125000,   13500,  162500,   92500,  158000,
          2000,  181000,  193000,  122358,   95200,   33033,   63063,
        207890,    7976,   16500,   99361,   33000,   80100,  160000,
         26000,   29000,   58000,   36000,  185000,  162000,   67000,
         52000,   68089,   58343,   38817,   56494,   79328,    5621,
         25538,   69779,   56290,    8500,    7032,  106000,  240000,
        214000,    1000,  265000,  134000,    2136,  250000,  130000,
         99500,  119000,   44665,  123000,   34000,  146000,   85000,
        201850,   46000,  190000,   43000,   14200,   19100,   54000,
        101500,   55735,    1500,  181491,   53319,   40906,   31711,
         43755,   66693,   59549,   94385,   73730,  221889,  116104,
        150546,  148120,  142000,  126000,   24000,  101000,  168000,
        165000,   98500,   98000,   77800,   51492,  136000,   91000,
        167000,  180000,    6000,   91500,  132478,   51000,  105000,
        137500,   88000,  156060,  107000,   57000,  300000,  200000,
         18945,  144000,  107825,    1620,  141000,  108000,   55000,
         37944,   90400,   96000,  193623,   63186,  219000,   23000,
         94000,  140500,  143000,  195000,  360003,    8079,  114368,
         79990,   81632,  155201,  101504,   90165,   86017,   85036,
        125531,  206000,   49700,   74000,    6550,   56000,   41779,
         31000,   13534,    7000,   28156,   63309,  155000,   11500,
          3000,  157000,   27000,   26300,   89100,  104000,  210000,
        216000,   84000,  161000,   72113,  113000,   96443,   51500,
        104500,   49102,   81000,  227000,  117000,    8576,   19723,
         21446,   52289,   52236,   54470,   25471,   33928,   59736,
         54290,    2118,   61379,   58544,   74381,   63982,   56429,
         19500,   56832,   30528,   65285,   20102,   52380,   74250,
         57247,  178000,   42323,   42462,   54723,   42545,   20375,
         43978,   40200,   27233,   28832,  103000,  142500,  115000,
         35500,   17500,   92000,    4500,   83000,    6750,   18500,
         50478,    9200,    2300,    4295,    5229,    4701,    6757,
         28182,   11533,   11688,   57728,   23712,   70670,   43381,
         79011,   58500,   43500,   12700,   49455,   46815,  334000,
        100875,  118000,  128000,  144030,  176000,   89000,   97000,
```

12800, 65100, 15200, 82000, 116500, 145000, 73000,
113226, 160500, 132000, 217000, 139000, 3500, 62000,
1500000, 120005, 16830, 93468, 16589, 53201, 25272,
53619, 6001, 202000, 71717, 84321, 177000, 205000,
41800, 88700, 188000, 225000, 5500, 4875, 2700,
58511, 26235, 26766, 1600, 24522, 13663, 58945,
375000, 156975, 27620, 143200, 138000, 189000, 156000,
10800, 67500, 255000, 71850, 112000, 159000, 76131,
51777, 64788, 46951, 74537, 23500, 68600, 13887,
66000, 24400, 27500, 67840, 380000, 112880, 61915,
184000, 6200, 16700, 7967, 9656, 68609, 33360,
14317, 87237, 62960, 26634, 163720, 67998, 20706,
100581, 67303, 79139, 84784, 130376, 33500, 22053,
78432, 270000, 7200, 117500, 78500, 115717, 4000,
26432, 46702, 46706, 53523, 56365, 49563, 56432,
52365, 53512, 46533, 58632, 65300, 39395, 230000,
32500, 64800, 248000, 66444, 218463, 153000, 87185,
137000, 3564, 48756, 80322, 173000, 148000, 6500,
220000, 108957, 22200, 260000, 42108, 79000, 17100,
43001, 100600, 92385, 116000, 72500, 82500, 50600,
121000, 101903, 85568, 3100, 186000, 59300, 89322,
48300, 23511, 22512, 62900, 52442, 34156, 34152,
62841, 38600, 27654, 48698, 32995, 90150, 85700,
25339, 41232, 68240, 92651, 88754, 23700, 69500,
28050, 22947, 7672, 36659, 44500, 9900, 124316,
6544, 49600, 198000, 103994, 35008, 31100, 42312,
54891, 18890, 75010, 4337, 14495, 83585, 48406,
56239, 33243, 15858, 87847, 87000, 112072, 52269,
15780, 2360457, 108800, 12500, 577414, 121941, 152186,
82246, 89580, 2789, 35278, 2860, 78562, 36088,
129000, 500000, 133000, 64500, 24500, 45500, 71500,
93500, 60300, 44391, 56315, 15151, 114321, 37333,
330000, 291000, 21500, 197000, 40300, 30030, 29029,
72072, 82082, 80600, 58609, 33003, 165500, 5800,
68700, 102996, 96500, 52412, 85472, 87452, 66530,
32331, 172000, 23456, 136500, 111000, 42500, 47200,
246000, 5200, 10500, 93331, 109000, 5400, 102000,
29700, 7600, 29340, 80800, 24265, 56900, 88200,
40800, 8588, 305000, 212000, 101200, 179150, 84487,
264000, 51146, 298000, 149000, 291977, 36800, 52200,
248200, 7720, 57882, 242000, 55380, 18816, 33019,
47747, 45900, 68697, 147279, 24700, 43526, 55885,
16034, 156040, 93415, 122000, 68519, 22966, 2350,
59872, 91182, 59500, 75500, 152500, 69123, 77524,
40523, 23600, 290000, 24177, 121779, 15381, 74800,
109322, 475000, 61000, 80500, 65755, 2600, 15500,
50700, 440000, 152000, 200400, 7300, 9750, 157138,
285000, 127991, 13120, 87500, 62200, 87540, 231438,
239451, 154000, 426000, 182000, 215000, 77300, 61260,
17601, 4773, 57900, 8600, 78010, 83844, 23999,
91400, 178500, 66953, 166000, 131000, 100750, 48676,
79500, 200185, 49025, 76460, 47370, 45217, 222300,
44600, 28080, 28800, 21900, 28100, 7673, 96272,
22500, 7400, 370000, 150360, 42130, 147500, 56194,
54188, 54043, 66657, 35582, 54327, 7949, 59734,
49185, 56389, 58245, 36422, 42535, 40736, 56246,
55403, 105358, 49200, 81500, 3177, 39414, 30078,
44588, 49907, 67082, 112879, 10200, 1300, 175802,
192000, 73840, 320000, 24857, 26442, 50800, 40142,
9400, 12584, 4300, 14548, 44772, 48500, 28782,
28451, 36500, 44885, 131111, 58400, 23400, 3010,
163000, 77088, 136511, 36710, 28180, 44077, 55768,
123278, 26500, 18484, 2560, 103655, 56975, 187000,
218000, 73257, 26320, 38426, 116700, 176062, 77500,
39500, 55896, 30400, 164000, 271000, 75262, 91863,
47552, 14700, 19700, 145241, 95500, 53473, 75958,
31596, 85710, 129627, 55130, 22522, 51856, 29434,
34500, 6825, 112011, 53534, 248119, 24019, 19600,
376412, 183000, 108916, 59865, 68140, 53190, 70100,

```
 47725,   70195,   77395,   74321,   80235,   77150,   61100,
 82300,   82050,   29899,   59235,  201000,   77215,   48228,
 58559,   50856,   18300,   59292,   21147,   61173,   91567,
 33400,   92686,   67600,   49500,  112048,   30646,   43011,
104300,   73500,  280000,   35700,   22700,   21871,   55425,
 37659,   37500,   19800,   45629,   68850,   45775,   68203,
 97343,   82507,   59400,  125876,   56800,   65204,    2286,
 15732,   49523,   36521,   64481,   49060,   37161,   60175,
 49957,   43235,   50699,       1,   50074,   30154,   26263,
 84925,   46357,   46737,  110048,  186388,   20171,   37151,
 58161,   50171,   55161,    9599,   16151,   28161,   80868,
194000,  191000])
```

- Essa variável "km_rodado" está formatado corretamente.

Variável 5 - "tipo_combustivel"

In [ ]:
```
#valores únicos:

carros.tipo_combustivel.unique()
```

Out[ ]:
```
array(['Diesel', 'Petrol', 'LPG', 'CNG'], dtype=object)
```

- Essa variável "tipo_combustivel" está formatado corretamente.

Variável 6 - "tipo_do_vendedor"

In [ ]:
```
#valores únicos:

carros.tipo_do_vendedor.unique()
```

Out[ ]:
```
array(['Individual', 'Dealer', 'Trustmark Dealer'], dtype=object)
```

- Essa variável "tipo_do_vendedor" está formatado corretamente.

Variável 7 - "transmissao"

In [ ]:
```
#valores únicos:

carros.transmissao.unique()
```

Out[ ]:
```
array(['Manual', 'Automatic'], dtype=object)
```

Essa variável "transmissao" está formatado corretamente.

Variável 8 - "dono"

In [ ]:
```
#valores únicos:

carros.dono.unique()
```

Out[ ]:
```
array(['First Owner', 'Second Owner', 'Third Owner',
       'Fourth & Above Owner', 'Test Drive Car'], dtype=object)
```

Essa variável "dono" está formatado corretamente.

Variável 9 - "consumo_do_combustivel"

In [ ]:
```
#valores únicos:

carros.consumo_do_combustivel.unique()
```

Out[ ]:
```
array(['23.4 kmpl', '21.14 kmpl', '17.7 kmpl', '23.0 kmpl', '16.1 kmpl',
       '20.14 kmpl', '17.3 km/kg', '23.59 kmpl', '20.0 kmpl',
       '19.01 kmpl', '17.3 kmpl', '19.3 kmpl', '18.9 kmpl', '18.15 kmpl',
       '24.52 kmpl', '19.7 kmpl', '22.54 kmpl', '21.0 kmpl', '25.5 kmpl',
       '26.59 kmpl', '21.5 kmpl', '20.3 kmpl', '21.4 kmpl', '24.7 kmpl',
       '18.2 kmpl', '16.8 kmpl', '24.3 kmpl', '14.0 kmpl', '18.6 kmpl',
       '33.44 km/kg', '23.95 kmpl', '17.0 kmpl', '20.63 kmpl',
       '13.93 kmpl', '16.0 kmpl', '17.8 kmpl', '18.5 kmpl', '12.55 kmpl',
       '12.99 kmpl', '14.8 kmpl', '13.5 kmpl', '26.0 kmpl', '20.65 kmpl',
       '27.3 kmpl', '11.36 kmpl', '17.68 kmpl', '14.28 kmpl',
       '18.53 kmpl', '14.84 kmpl', '21.12 kmpl', '20.36 kmpl',
       '21.27 kmpl', '18.16 kmpl', '22.0 kmpl', '25.1 kmpl', '20.51 kmpl',
       '21.66 kmpl', '25.2 kmpl', '22.9 kmpl', '16.02 kmpl', '20.54 kmpl',
       '22.77 kmpl', '15.71 kmpl', '23.1 kmpl', '19.02 kmpl',
       '19.81 kmpl', '26.2 km/kg', '16.47 kmpl', '15.04 kmpl',
       '19.1 kmpl', '21.79 kmpl', '18.8 kmpl', '21.21 kmpl', '15.37 kmpl',
       '11.79 kmpl', '19.0 kmpl', '14.3 kmpl', '15.8 kmpl', '15.1 kmpl',
       '19.09 kmpl', '22.32 kmpl', '21.9 kmpl', '14.53 kmpl',
       '21.63 kmpl', '20.85 kmpl', '20.45 kmpl', '19.67 kmpl',
       '23.01 kmpl', '20.77 kmpl', '17.92 kmpl', '17.01 kmpl',
       '22.37 kmpl', '19.33 kmpl', '9.5 kmpl', '12.83 kmpl', '22.48 kmpl',
       '16.78 kmpl', '14.67 kmpl', '15.0 kmpl', '13.96 kmpl', '18.0 kmpl',
       '12.07 kmpl', '26.21 kmpl', '10.8 kmpl', '16.3 kmpl', '13.6 kmpl',
       '14.74 kmpl', '15.6 kmpl', '19.56 kmpl', '22.69 kmpl',
       '19.16 kmpl', '18.12 kmpl', '12.1 kmpl', '17.5 kmpl', '42.0 kmpl',
       '20.4 kmpl', '21.1 kmpl', '19.44 kmpl', '13.0 kmpl', '21.43 kmpl',
       '22.95 kmpl', '16.2 kmpl', '15.3 kmpl', '28.09 kmpl', '17.4 kmpl',
       '19.4 kmpl', '26.6 km/kg', '17.6 kmpl', '28.4 kmpl', '14.1 kmpl',
       '25.17 kmpl', '22.74 kmpl', '17.57 kmpl', '16.95 kmpl',
       '19.49 kmpl', '17.21 kmpl', '13.2 kmpl', '14.2 kmpl', '26.8 kmpl',
       '25.4 kmpl', '11.5 kmpl', '27.28 kmpl', '17.97 kmpl', '12.8 kmpl',
       '16.55 kmpl', '12.05 kmpl', '14.07 kmpl', '21.02 kmpl',
       '11.57 kmpl', '17.9 kmpl', '15.96 kmpl', '17.1 kmpl', '17.19 kmpl',
       '21.01 kmpl', '24.0 kmpl', '25.6 kmpl', '21.38 kmpl', '23.84 kmpl',
       '23.08 kmpl', '14.24 kmpl', '20.71 kmpl', '15.64 kmpl',
       '14.5 kmpl', '16.34 kmpl', '27.39 kmpl', '11.1 kmpl', '13.9 kmpl',
       '20.88 km/kg', '20.92 kmpl', '23.8 kmpl', '24.4 kmpl',
       '15.29 kmpl', '21.19 kmpl', '22.5 kmpl', '19.6 kmpl', '23.65 kmpl',
       '25.32 kmpl', '23.5 kmpl', '16.6 kmpl', '23.9 kmpl', '20.8 kmpl',
       '27.62 kmpl', '12.9 kmpl', '25.44 kmpl', '17.88 kmpl', '22.7 kmpl',
       '17.2 kmpl', '15.42 kmpl', '19.68 kmpl', '18.7 kmpl', '15.4 kmpl',
       '19.34 kmpl', '22.71 kmpl', '25.8 kmpl', '13.7 kmpl', '12.2 kmpl',
       '18.49 kmpl', '9.0 kmpl', '0.0 kmpl', '13.58 kmpl', '10.1 kmpl',
       '20.5 kmpl', '25.0 kmpl', '10.5 kmpl', '22.07 kmpl', '22.3 kmpl',
       '15.26 kmpl', '20.62 kmpl', '27.4 kmpl', '23.2 kmpl', '14.4 kmpl',
       '18.4 kmpl', '30.46 km/kg', '14.02 kmpl', '11.0 kmpl', '20.6 kmpl',
       '22.05 kmpl', '20.2 kmpl', '18.1 kmpl', '22.1 kmpl', '19.87 kmpl',
       '13.01 kmpl', '18.06 kmpl', '26.1 kmpl', '16.52 kmpl',
       '13.55 kmpl', '24.2 kmpl', '25.83 kmpl', '11.2 kmpl', '17.09 kmpl',
       '21.03 kmpl', '17.45 kmpl', '21.64 kmpl', '21.94 km/kg',
       '13.87 kmpl', '19.98 kmpl', '20.52 kmpl', '23.57 kmpl',
       '11.7 kmpl', '17.43 kmpl', '18.88 kmpl', '13.68 kmpl',
       '11.18 kmpl', '20.89 kmpl', '11.8 kmpl', '19.62 kmpl', '21.7 kmpl',
       '14.9 kmpl', '19.5 kmpl', '10.91 kmpl', '15.7 kmpl', '20.73 kmpl',
       '15.85 kmpl', '20.7 kmpl', '14.23 kmpl', '16.5 kmpl', '17.36 kmpl',
       '12.6 kmpl', '16.36 kmpl', '14.95 kmpl', '16.9 kmpl', '19.2 kmpl',
       '16.96 kmpl', '22.15 kmpl', '18.78 kmpl', '19.61 kmpl',
       '17.71 kmpl', '18.3 kmpl', '19.12 kmpl', '19.72 kmpl', '12.0 kmpl',
       '11.4 kmpl', '23.03 kmpl', '11.07 kmpl', '15.9 kmpl', '17.67 kmpl',
       '20.46 kmpl', '13.1 kmpl', '13.45 km/kg', '24.8 kmpl',
       '15.73 kmpl', '15.11 kmpl', '12.7 kmpl', '21.2 kmpl', '20.38 kmpl',
```

```
 '21.56 kmpl', '13.22 kmpl', '14.49 kmpl', '15.05 kmpl',
 '23.26 kmpl', '15.41 kmpl', '13.8 kmpl', '22.27 kmpl',
 '32.52 km/kg', '14.66 kmpl', '12.12 kmpl', '16.84 kmpl',
 '14.09 kmpl', '14.7 kmpl', '13.4 kmpl', '15.5 kmpl', '13.49 kmpl',
 '11.88 km/kg', '14.6 kmpl', '10.75 kmpl', '24.5 kmpl',
 '11.74 kmpl', '16.07 kmpl', '15.63 kmpl', '26.3 km/kg',
 '23.7 km/kg', '25.47 kmpl', '17.05 kmpl', '23.3 kmpl', '11.9 kmpl',
 '13.38 kmpl', '20.86 kmpl', '19.2 km/kg', '10.9 kmpl',
 '18.25 kmpl', '15.2 kmpl', '20.37 kmpl', '17.8 km/kg', '21.8 kmpl',
 '11.96 kmpl', '24.04 kmpl', '19.69 kmpl', '13.73 kmpl',
 '21.04 kmpl', '25.01 kmpl', '10.93 kmpl', '10.9 km/kg',
 '24.29 kmpl', '13.44 kmpl', '20.07 kmpl', '21.1 km/kg',
 '19.08 kmpl', '20.34 kmpl', '11.68 kmpl', '12.5 kmpl', '12.3 kmpl',
 '23.87 kmpl', '16.38 kmpl', '17.42 kmpl', '10.0 kmpl',
 '18.24 kmpl', '10.71 kmpl', '19.59 kmpl', '16.7 kmpl',
 '19.83 kmpl', '21.76 kmpl', '16.05 kmpl', '20.28 kmpl',
 '16.25 kmpl', '16.73 kmpl', '18.48 kmpl', '13.2 km/kg',
 '21.4 km/kg', '14.99 kmpl', '18.76 kmpl', '16.4 kmpl',
 '19.64 kmpl', '14.94 kmpl', '16.6 km/kg', '16.0 km/kg',
 '17.11 kmpl', '22.8 km/kg', '32.26 km/kg', '33.0 km/kg',
 '12.4 kmpl', '18.44 kmpl', '16.09 kmpl', '19.0 km/kg',
 '12.62 kmpl', '21.13 kmpl', '15.17 kmpl', '21.73 kmpl',
 '21.72 kmpl', '12.85 kmpl', '14.81 kmpl', '13.24 kmpl',
 '14.4 km/kg', '21.49 kmpl', '14.62 kmpl', '26.83 km/kg',
 '11.45 kmpl', '12.08 kmpl', '15.74 kmpl', '11.3 kmpl',
 '15.1 km/kg', '14.21 kmpl', '11.72 kmpl', '16.51 kmpl'],
dtype=object)
```

Observação:

Essa variável tem um grande problema a ser resolvido. Isso porque as unidades de alguns valores numéricos variam. Ao notar o resultado da função acima identifiquei que existem as seguintes unidades para "consumo_do_combustivel":

- 1 - kmpl
- 2 - km/kg

Pela literatura* notei que não há como apenas converter uma variável pela outra, já que é necessária saber a pressão exata em que o gás é pressurizado no veículo, para assim, poder realizar a conversão.

*Fonte da referencia:

https://datascience.stackexchange.com/questions/55663/one-feature-several-units

https://math.stackexchange.com/questions/1141752/how-to-convert-kilometers-kilogram-km-kg-to-miles-gallon-mpg

Como não disponho dessas informações, para esse projeto em particular irei apenas **trabalhar com os valores que estão em "kmpl"**, ou seja, kilometros rodados por litro.

In [ ]:
```
# Vou remover as unidades de uma lista de valores

# Usando replace() + strip() + list comprehension

import re
```

In [ ]:
```
# criando uma nova lista

test_list = carros.consumo_do_combustivel
```

In [ ]:
```

```

```
# Imprimindo a lista original
print("A lista original é : " + str(test_list))
```

```
A lista original é : 0        23.4 kmpl
1       21.14 kmpl
2        17.7 kmpl
3        23.0 kmpl
4        16.1 kmpl
           ...
8123     18.5 kmpl
8124     16.8 kmpl
8125     19.3 kmpl
8126    23.57 kmpl
8127    23.57 kmpl
Name: consumo_do_combustivel, Length: 7906, dtype: object
```

In [ ]:
```
# Inicializando a unidade
unit = "kmpl"
```

In [ ]:
```
# Removendo unidades de uma lista de valores
# Usando replace() + strip() + list comprehension
carros.consumo_do_combustivel = [sub.replace(unit, "").strip() for sub in test_list]
```

In [ ]:
```
# Imprimindo o resultado
print("Lista após a remoção das unidades : " + str(carros.consumo_do_combustivel))
```

```
Lista após a remoção das unidades : 0        23.4
1       21.14
2        17.7
3        23.0
4        16.1
         ...
8123     18.5
8124     16.8
8125     19.3
8126    23.57
8127    23.57
Name: consumo_do_combustivel, Length: 7906, dtype: object
```

In [ ]:
```
#Checando no dataset a variável "consumo_do_combustivel"

carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda_USD	kilometragem	tipo_combustivel	tipo_do_
0	Maruti Swift Dzire VDI	2014	6164.38	145500	Diesel	
1	Skoda Rapid 1.5 TDI Ambition	2014	5068.49	120000	Diesel	
2	Honda City 2017-2020 EXi	2006	2164.38	140000	Petrol	
3	Hyundai i20 Sportz Diesel	2010	3082.19	127000	Diesel	
4	Maruti Swift VXI BSIII	2007	1780.82	120000	Petrol	

Vou ficar apenas com os valores em "kmpl"

In [ ]:
```
#Removendo os valores que não são numéricos da variável "consumo_do_combustivel"

carros = carros[pd.to_numeric(carros['consumo_do_combustivel'], errors='coerce').notr

carros.head()
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda_USD	kilometragem	tipo_combustivel	tipo_do_
0	Maruti Swift Dzire VDI	2014	6164.38	145500	Diesel	
1	Skoda Rapid 1.5 TDI Ambition	2014	5068.49	120000	Diesel	
2	Honda City 2017-2020 EXi	2006	2164.38	140000	Petrol	
3	Hyundai i20 Sportz Diesel	2010	3082.19	127000	Diesel	
4	Maruti Swift VXI BSIII	2007	1780.82	120000	Petrol	

In [ ]:
```
#Checando se os valores estão todos sem unidades

carros.consumo_do_combustivel.unique()
```

Out[ ]:
```
array(['23.4', '21.14', '17.7', '23.0', '16.1', '20.14', '23.59', '20.0',
       '19.01', '17.3', '19.3', '18.9', '18.15', '24.52', '19.7', '22.54',
       '21.0', '25.5', '26.59', '21.5', '20.3', '21.4', '24.7', '18.2',
       '16.8', '24.3', '14.0', '18.6', '23.95', '17.0', '20.63', '13.93',
       '16.0', '17.8', '18.5', '12.55', '12.99', '14.8', '13.5', '26.0',
       '20.65', '27.3', '11.36', '17.68', '14.28', '18.53', '14.84',
       '21.12', '20.36', '21.27', '18.16', '22.0', '25.1', '20.51',
       '21.66', '25.2', '22.9', '16.02', '20.54', '22.77', '15.71',
       '23.1', '19.02', '19.81', '16.47', '15.04', '19.1', '21.79',
       '18.8', '21.21', '15.37', '11.79', '19.0', '14.3', '15.8', '15.1',
       '19.09', '22.32', '21.9', '14.53', '21.63', '20.85', '20.45',
       '19.67', '23.01', '20.77', '17.92', '17.01', '22.37', '19.33',
       '9.5', '12.83', '22.48', '16.78', '14.67', '15.0', '13.96', '18.0',
       '12.07', '26.21', '10.8', '16.3', '13.6', '14.74', '15.6', '19.56',
       '22.69', '19.16', '18.12', '12.1', '17.5', '42.0', '20.4', '21.1',
       '19.44', '13.0', '21.43', '22.95', '16.2', '15.3', '28.09', '17.4',
       '19.4', '17.6', '28.4', '14.1', '25.17', '22.74', '17.57', '16.95',
       '19.49', '17.21', '13.2', '14.2', '26.8', '25.4', '11.5', '27.28',
       '17.97', '12.8', '16.55', '12.05', '14.07', '21.02', '11.57',
       '17.9', '15.96', '17.1', '17.19', '21.01', '24.0', '25.6', '21.38',
       '23.84', '23.08', '14.24', '20.71', '15.64', '14.5', '16.34',
       '27.39', '11.1', '13.9', '20.92', '23.8', '24.4', '15.29', '21.19',
       '22.5', '19.6', '23.65', '25.32', '23.5', '16.6', '23.9', '20.8',
       '27.62', '12.9', '25.44', '17.88', '22.7', '17.2', '15.42',
       '19.68', '18.7', '15.4', '19.34', '22.71', '25.8', '13.7', '12.2',
       '18.49', '9.0', '0.0', '13.58', '10.1', '20.5', '25.0', '10.5',
       '22.07', '22.3', '15.26', '20.62', '27.4', '23.2', '14.4', '18.4',
       '14.02', '11.0', '20.6', '22.05', '20.2', '18.1', '22.1', '19.87',
       '13.01', '18.06', '26.1', '16.52', '13.55', '24.2', '25.83',
       '11.2', '17.09', '21.03', '17.45', '21.64', '13.87', '19.98',
       '20.52', '23.57', '11.7', '17.43', '18.88', '13.68', '11.18',
       '20.89', '11.8', '19.62', '21.7', '14.9', '19.5', '10.91', '15.7',
       '20.73', '15.85', '20.7', '14.23', '16.5', '17.36', '12.6',
       '16.36', '14.95', '16.9', '19.2', '16.96', '22.15', '18.78',
       '19.61', '17.71', '18.3', '19.12', '19.72', '12.0', '11.4',
       '23.03', '11.07', '15.9', '17.67', '20.46', '13.1', '24.8',
       '15.73', '15.11', '12.7', '21.2', '20.38', '21.56', '13.22',
       '14.49', '15.05', '23.26', '15.41', '13.8', '22.27', '14.66',
       '12.12', '16.84', '14.09', '14.7', '13.4', '15.5', '13.49', '14.6',
```

```
'10.75', '24.5', '11.74', '16.07', '15.63', '25.47', '17.05',
'23.3', '11.9', '13.38', '20.86', '10.9', '18.25', '15.2', '20.37',
'21.8', '11.96', '24.04', '19.69', '13.73', '21.04', '25.01',
'10.93', '24.29', '13.44', '20.07', '19.08', '20.34', '11.68',
'12.5', '12.3', '23.87', '16.38', '17.42', '10.0', '18.24',
'10.71', '19.59', '16.7', '19.83', '21.76', '16.05', '20.28',
'16.25', '16.73', '18.48', '14.99', '18.76', '16.4', '19.64',
'14.94', '17.11', '12.4', '18.44', '16.09', '12.62', '21.13',
'15.17', '21.73', '21.72', '12.85', '14.81', '13.24', '21.49',
'14.62', '11.45', '12.08', '15.74', '11.3', '14.21', '11.72',
'16.51'], dtype=object)
```

Observei pela função acima que apesar de não ter mais nenhuma unidade nessa variável, ela ainda está como "object". Preciso alterar para tipo numérico.

In [ ]:
```
#Mudando o tipo da variável

carros["consumo_do_combustivel"]=carros["consumo_do_combustivel"].astype(float)

carros.consumo_do_combustivel.unique()
```

Out[ ]:
```
array([23.4 , 21.14, 17.7 , 23.  , 16.1 , 20.14, 23.59, 20.  , 19.01,
       17.3 , 19.3 , 18.9 , 18.15, 24.52, 19.7 , 22.54, 21.  , 25.5 ,
       26.59, 21.5 , 20.3 , 21.4 , 24.7 , 18.2 , 16.8 , 24.3 , 14.  ,
       18.6 , 23.95, 17.  , 20.63, 13.93, 16.  , 17.8 , 18.5 , 12.55,
       12.99, 14.8 , 13.5 , 26.  , 20.65, 27.3 , 11.36, 17.68, 14.28,
       18.53, 14.84, 21.12, 20.36, 21.27, 18.16, 22.  , 25.1 , 20.51,
       21.66, 25.2 , 22.9 , 16.02, 20.54, 22.77, 15.71, 23.1 , 19.02,
       19.81, 16.47, 15.04, 19.1 , 21.79, 18.8 , 21.21, 15.37, 11.79,
       19.  , 14.3 , 15.8 , 15.1 , 19.09, 22.32, 21.9 , 14.53, 21.63,
       20.85, 20.45, 19.67, 23.01, 20.77, 17.92, 17.01, 22.37, 19.33,
        9.5 , 12.83, 22.48, 16.78, 14.67, 15.  , 13.96, 18.  , 12.07,
       26.21, 10.8 , 16.3 , 13.6 , 14.74, 15.6 , 19.56, 22.69, 19.16,
       18.12, 12.1 , 17.5 , 42.  , 20.4 , 21.1 , 19.44, 13.  , 21.43,
       22.95, 16.2 , 15.3 , 28.09, 17.4 , 19.4 , 17.6 , 28.4 , 14.1 ,
       25.17, 22.74, 17.57, 16.95, 19.49, 17.21, 13.2 , 14.2 , 26.8 ,
       25.4 , 11.5 , 27.28, 17.97, 12.8 , 16.55, 12.05, 14.07, 21.02,
       11.57, 17.9 , 15.96, 17.1 , 17.19, 21.01, 24.  , 25.6 , 21.38,
       23.84, 23.08, 14.24, 20.71, 15.64, 14.5 , 16.34, 27.39, 11.1 ,
       13.9 , 20.92, 23.8 , 24.4 , 15.29, 21.19, 22.5 , 19.6 , 23.65,
       25.32, 23.5 , 16.6 , 23.9 , 20.8 , 27.62, 12.9 , 25.44, 17.88,
       22.7 , 17.2 , 15.42, 19.68, 18.7 , 15.4 , 19.34, 22.71, 25.8 ,
       13.7 , 12.2 , 18.49,  9.  ,  0.  , 13.58, 10.1 , 20.5 , 25.  ,
       10.5 , 22.07, 22.3 , 15.26, 20.62, 27.4 , 23.2 , 14.4 , 18.4 ,
       14.02, 11.  , 20.6 , 22.05, 20.2 , 18.1 , 22.1 , 19.87, 13.01,
       18.06, 26.1 , 16.52, 13.55, 24.2 , 25.83, 11.2 , 17.09, 21.03,
       17.45, 21.64, 13.87, 19.98, 20.52, 23.57, 11.7 , 17.43, 18.88,
       13.68, 11.18, 20.89, 11.8 , 19.62, 21.7 , 14.9 , 19.5 , 10.91,
       15.7 , 20.73, 15.85, 20.7 , 14.23, 16.5 , 17.36, 12.6 , 16.36,
       14.95, 16.9 , 19.2 , 16.96, 22.15, 18.78, 19.61, 17.71, 18.3 ,
       19.12, 19.72, 12.  , 11.4 , 23.03, 11.07, 15.9 , 17.67, 20.46,
       13.1 , 24.8 , 15.73, 15.11, 12.7 , 21.2 , 20.38, 21.56, 13.22,
       14.49, 15.05, 23.26, 15.41, 13.8 , 22.27, 14.66, 12.12, 16.84,
       14.09, 14.7 , 13.4 , 15.5 , 13.49, 14.6 , 10.75, 24.5 , 11.74,
       16.07, 15.63, 25.47, 17.05, 23.3 , 11.9 , 13.38, 20.86, 10.9 ,
       18.25, 15.2 , 20.37, 21.8 , 11.96, 24.04, 19.69, 13.73, 21.04,
       25.01, 10.93, 24.29, 13.44, 20.07, 19.08, 20.34, 11.68, 12.5 ,
       12.3 , 23.87, 16.38, 17.42, 10.  , 18.24, 10.71, 19.59, 16.7 ,
       19.83, 21.76, 16.05, 20.28, 16.25, 16.73, 18.48, 14.99, 18.76,
       16.4 , 19.64, 14.94, 17.11, 12.4 , 18.44, 16.09, 12.62, 21.13,
       15.17, 21.73, 21.72, 12.85, 14.81, 13.24, 21.49, 14.62, 11.45,
       12.08, 15.74, 11.3 , 14.21, 11.72, 16.51])
```

In [ ]:
```
#Checando se a variável está como tipo numérico

carros.dtypes
```

Out[ ]:
```
nome_completo_do_carro     object
ano_de_venda                int64
preco_de_venda_USD        float64
kilometragem                int64
tipo_combustivel           object
tipo_do_vendedor           object
transmissao                object
dono                       object
consumo_do_combustivel    float64
motor                      object
potencia_do_motor          object
torque                     object
assentos                  float64
Marca                      object
dtype: object
```

Agora a variável está com o tipo de valores corretos. Falta apenas colocar no título da variável a unidade "kmpl"

In [ ]:
```
#Trocando o título da variável

carros.rename(columns={"consumo_do_combustivel": "consumo_do_combustivel_kmpl"}, inpl

carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda_USD	kilometragem	tipo_combustivel	tipo_do_
0	Maruti Swift Dzire VDI	2014	6164.38	145500	Diesel	
1	Skoda Rapid 1.5 TDI Ambition	2014	5068.49	120000	Diesel	
2	Honda City 2017-2020 EXi	2006	2164.38	140000	Petrol	
3	Hyundai i20 Sportz Diesel	2010	3082.19	127000	Diesel	
4	Maruti Swift VXI BSIII	2007	1780.82	120000	Petrol	

Agora sim esta variável "consumo_do_combustivel" está devidamente formatada para as análises.

Variável 10 - "motor"

In [ ]:
```
#valores únicos:

carros.motor.unique()
```

Out[ ]:
```
array(['1248 CC', '1498 CC', '1497 CC', '1396 CC', '1298 CC', '1197 CC',
       '796 CC', '1364 CC', '1399 CC', '1461 CC', '993 CC', '1061 CC',
       '1198 CC', '1199 CC', '998 CC', '1591 CC', '2179 CC', '1368 CC',
       '2982 CC', '2494 CC', '2143 CC', '2477 CC', '1462 CC', '2755 CC',
       '1968 CC', '1798 CC', '1196 CC', '1373 CC', '1598 CC', '1998 CC',
       '1086 CC', '1194 CC', '1172 CC', '1405 CC', '1582 CC', '999 CC',
       '2487 CC', '1999 CC', '3604 CC', '2987 CC', '1995 CC', '1451 CC',
       '1969 CC', '2967 CC', '2497 CC', '1797 CC', '1991 CC', '2362 CC',
       '1493 CC', '1599 CC', '1341 CC', '1794 CC', '799 CC', '1193 CC',
       '2696 CC', '1495 CC', '1186 CC', '1047 CC', '2498 CC', '2956 CC',
       '2523 CC', '1120 CC', '624 CC', '1496 CC', '1984 CC', '2354 CC',
       '814 CC', '793 CC', '1799 CC', '936 CC', '1956 CC', '1997 CC',
       '1499 CC', '1948 CC', '2997 CC', '2489 CC', '2499 CC', '2609 CC',
       '2953 CC', '1150 CC', '1994 CC', '1388 CC', '1527 CC', '2199 CC',
```

```
'995 CC', '2993 CC', '1586 CC', '1390 CC', '909 CC', '2393 CC',
'3198 CC', '1339 CC', '2835 CC', '2092 CC', '1595 CC', '2496 CC',
'1596 CC', '1597 CC', '2596 CC', '2148 CC', '1299 CC', '1590 CC',
'2231 CC', '2694 CC', '2200 CC', '1795 CC', '1896 CC', '1796 CC',
'1422 CC', '1489 CC', '2359 CC', '2197 CC', '2999 CC', '1781 CC',
'2650 CC', '1343 CC', '2446 CC', '3498 CC', '2198 CC', '2776 CC',
'1950 CC'], dtype=object)
```

- Notei que esses valores estão com uma unidade chamada de "CC". Essa unidade pode vir descrita apenas na legenda dessa variável.

- Assim, irei remover essa unidade dos respectivos valores e irei colocar na legenda a unidade "CC".

In [ ]:
```
# Removendo as unidades de uma lista de valores
# Usando replace() + strip() + list comprehension
import re
```

In [ ]:
```
# Iniciando uma nova lista
test_list = carros.motor
```

In [ ]:
```
# Imprimindo a lista original
print("A lista original é : " + str(test_list))
```

```
A lista original é : 0       1248 CC
1       1498 CC
2       1497 CC
3       1396 CC
4       1298 CC
        ...
8123    1197 CC
8124    1493 CC
8125    1248 CC
8126    1396 CC
8127    1396 CC
Name: motor, Length: 7819, dtype: object
```

In [ ]:
```
# Inicializando a unidade
unit = "CC"
```

In [ ]:
```
# Removendo as unidades de uma lista de valores
# Usando replace() + strip() + list comprehension
carros.motor = [sub.replace(unit, "").strip() for sub in test_list]
```

In [ ]:
```
# Imprimindo o resultado
print("Lista após a remoção das unidades : " + str(carros.motor))
```

```
Lista após a remoção das unidades : 0       1248
1       1498
2       1497
3       1396
4       1298
        ...
8123    1197
8124    1493
8125    1248
8126    1396
8127    1396
Name: motor, Length: 7819, dtype: object
```

In [ ]:
```
#valores únicos:
```

```
carros.motor.unique()
```

Out[ ]:
```
array(['1248', '1498', '1497', '1396', '1298', '1197', '796', '1364',
       '1399', '1461', '993', '1061', '1198', '1199', '998', '1591',
       '2179', '1368', '2982', '2494', '2143', '2477', '1462', '2755',
       '1968', '1798', '1196', '1373', '1598', '1998', '1086', '1194',
       '1172', '1405', '1582', '999', '2487', '1999', '3604', '2987',
       '1995', '1451', '1969', '2967', '2497', '1797', '1991', '2362',
       '1493', '1599', '1341', '1794', '799', '1193', '2696', '1495',
       '1186', '1047', '2498', '2956', '2523', '1120', '624', '1496',
       '1984', '2354', '814', '793', '1799', '936', '1956', '1997',
       '1499', '1948', '2997', '2489', '2499', '2609', '2953', '1150',
       '1994', '1388', '1527', '2199', '995', '2993', '1586', '1390',
       '909', '2393', '3198', '1339', '2835', '2092', '1595', '2496',
       '1596', '1597', '2596', '2148', '1299', '1590', '2231', '2694',
       '2200', '1795', '1896', '1796', '1422', '1489', '2359', '2197',
       '2999', '1781', '2650', '1343', '2446', '3498', '2198', '2776',
       '1950'], dtype=object)
```

In [ ]:
```
#Checando se o tipo da variável está como numérica

carros.dtypes
```

Out[ ]:
```
nome_completo_do_carro          object
ano_de_venda                     int64
preco_de_venda_USD             float64
kilometragem                     int64
tipo_combustivel                object
tipo_do_vendedor                object
transmissao                     object
dono                            object
consumo_do_combustivel_kmpl    float64
motor                           object
potencia_do_motor               object
torque                          object
assentos                       float64
Marca                           object
dtype: object
```

Ainda está como "object". Terei de trocar por tipo de valor numérico.

In [ ]:
```
carros["motor"]=carros["motor"].astype(float)

carros.motor.unique()
```

Out[ ]:
```
array([1248., 1498., 1497., 1396., 1298., 1197.,  796., 1364., 1399.,
       1461.,  993., 1061., 1198., 1199.,  998., 1591., 2179., 1368.,
       2982., 2494., 2143., 2477., 1462., 2755., 1968., 1798., 1196.,
       1373., 1598., 1998., 1086., 1194., 1172., 1405., 1582.,  999.,
       2487., 1999., 3604., 2987., 1995., 1451., 1969., 2967., 2497.,
       1797., 1991., 2362., 1493., 1599., 1341., 1794.,  799., 1193.,
       2696., 1495., 1186., 1047., 2498., 2956., 2523., 1120.,  624.,
       1496., 1984., 2354.,  814.,  793., 1799.,  936., 1956., 1997.,
       1499., 1948., 2997., 2489., 2499., 2609., 2953., 1150., 1994.,
       1388., 1527., 2199.,  995., 2993., 1586., 1390.,  909., 2393.,
       3198., 1339., 2835., 2092., 1595., 2496., 1596., 1597., 2596.,
       2148., 1299., 1590., 2231., 2694., 2200., 1795., 1896., 1796.,
       1422., 1489., 2359., 2197., 2999., 1781., 2650., 1343., 2446.,
       3498., 2198., 2776., 1950.])
```

In [ ]:
```
#Identificando os valores e o tipo da variável

carros.motor
```

Out[ ]:
```
0       1248.0
1       1498.0
```

```
2       1497.0
3       1396.0
4       1298.0
         ...
8123    1197.0
8124    1493.0
8125    1248.0
8126    1396.0
8127    1396.0
Name: motor, Length: 7819, dtype: float64
```

Agora sim essa variável "motor" está com o tipo correto. Falta apenas colocar a unidade no título dessa variável.

In [ ]:
```
#Trocando o título da variável

carros.rename(columns={"motor": "motor_CC"}, inplace=True)

carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda_USD	kilometragem	tipo_combustivel	tipo_do
0	Maruti Swift Dzire VDI	2014	6164.38	145500	Diesel	
1	Skoda Rapid 1.5 TDI Ambition	2014	5068.49	120000	Diesel	
2	Honda City 2017-2020 EXi	2006	2164.38	140000	Petrol	
3	Hyundai i20 Sportz Diesel	2010	3082.19	127000	Diesel	
4	Maruti Swift VXI BSIII	2007	1780.82	120000	Petrol	

Agora sim esta variável "motor" está devidamente formatada.

Variável 11 - "potencia_do_motor"

In [ ]:
```
#valores únicos:

carros.potencia_do_motor.unique()
```

Out[ ]:
```
array(['74 bhp', '103.52 bhp', '78 bhp', '90 bhp', '88.2 bhp',
       '81.86 bhp', '37 bhp', '67.1 bhp', '68.1 bhp', '108.45 bhp',
       '60 bhp', '73.9 bhp', '67 bhp', '82 bhp', '88.5 bhp', '46.3 bhp',
       '88.73 bhp', '64.1 bhp', '98.6 bhp', '88.8 bhp', '83.81 bhp',
       '83.1 bhp', '47.3 bhp', '73.8 bhp', '34.2 bhp', '35 bhp',
       '81.83 bhp', '121.3 bhp', '138.03 bhp', '160.77 bhp', '117.3 bhp',
       '116.3 bhp', '83.14 bhp', '67.05 bhp', '168.5 bhp', '100 bhp',
       '120.7 bhp', '98.63 bhp', '175.56 bhp', '103.25 bhp', '171.5 bhp',
       '100.6 bhp', '174.33 bhp', '187.74 bhp', '170 bhp', '78.9 bhp',
       '88.76 bhp', '86.8 bhp', '108.495 bhp', '108.62 bhp', '93.7 bhp',
       '103.6 bhp', '98.59 bhp', '189 bhp', '67.04 bhp', '68.05 bhp',
       '82.85 bhp', '81.80 bhp', '73 bhp', '120 bhp', '94.68 bhp',
       '160 bhp', '65 bhp', '155 bhp', '69.01 bhp', '126.32 bhp',
       '138.1 bhp', '83.8 bhp', '126.2 bhp', '98.96 bhp', '62.1 bhp',
       '86.7 bhp', '188 bhp', '214.56 bhp', '177 bhp', '280 bhp',
       '148.31 bhp', '254.79 bhp', '190 bhp', '177.46 bhp', '204 bhp',
       '141 bhp', '117.6 bhp', '241.4 bhp', '282 bhp', '150 bhp',
       '147.5 bhp', '108.5 bhp', '103.5 bhp', '183 bhp', '181.04 bhp',
       '157.7 bhp', '164.7 bhp', '91.1 bhp', '400 bhp', '68 bhp',
```

```
'75 bhp', '85.8 bhp', '87.2 bhp', '53 bhp', '118 bhp', '103.2 bhp',
'83 bhp', '84 bhp', '147.94 bhp', '74.02 bhp', '53.3 bhp',
'80 bhp', '88.7 bhp', '97.7 bhp', '121.36 bhp', '162 bhp',
'140 bhp', '94 bhp', '100.57 bhp', '82.9 bhp', '83.11 bhp',
'70 bhp', '153.86 bhp', '121 bhp', '126.3 bhp', '73.97 bhp',
'171 bhp', '69 bhp', '99.6 bhp', '102 bhp', '105 bhp', '63 bhp',
'79.4 bhp', '97.9 bhp', '63.1 bhp', '66.1 bhp', '110 bhp',
'174.5 bhp', '53.26 bhp', '73.75 bhp', '67.06 bhp', '64.08 bhp',
'37.5 bhp', '189.3 bhp', '158.8 bhp', '55.2 bhp', '71.01 bhp',
'73.74 bhp', '147.9 bhp', '71 bhp', '77 bhp', '121.4 bhp',
'113.4 bhp', '47 bhp', '130 bhp', '57.6 bhp', '138 bhp',
'52.8 bhp', '53.64 bhp', '53.5 bhp', '76.8 bhp', '82.4 bhp',
'113.42 bhp', '76 bhp', '84.8 bhp', '56.3 bhp', '218 bhp',
'112 bhp', '92 bhp', '105.5 bhp', '169 bhp', '95 bhp', '72.4 bhp',
'115 bhp', '152 bhp', '91.2 bhp', '156 bhp', '74.9 bhp', '62 bhp',
'105.3 bhp', '73.94 bhp', '85.80 bhp', '85 bhp', '118.3 bhp',
'72 bhp', '147.51 bhp', '58 bhp', '64 bhp', '126.24 bhp',
'76.9 bhp', '194.3 bhp', '99.23 bhp', '89.84 bhp', '123.7 bhp',
'118.35 bhp', '99 bhp', '241 bhp', '136 bhp', '261.4 bhp',
'104.68 bhp', '37.48 bhp', '104 bhp', '88.50 bhp', '63.12 bhp',
'91.7 bhp', '102.5 bhp', '177.6 bhp', '45 bhp', '123.37 bhp',
'147.8 bhp', '184 bhp', '84.48 bhp', '68.07 bhp', '74.96 bhp',
'167.6 bhp', '152.87 bhp', '112.2 bhp', '83.83 bhp', '197 bhp',
'110.4 bhp', '104.55 bhp', '103 bhp', '103.3 bhp', '66 bhp',
'108.6 bhp', '165 bhp', '163.7 bhp', '116.9 bhp', '94.93 bhp',
'127 bhp', '198.5 bhp', '179.5 bhp', '120.69 bhp', '121.31 bhp',
'138.08 bhp', '187.7 bhp', '80.8 bhp', '86.79 bhp', '93.87 bhp',
'116.6 bhp', '143 bhp', '92.7 bhp', '88 bhp', '78.8 bhp',
'64.4 bhp', '125 bhp', '139.01 bhp', '254.8 bhp', '181 bhp',
'258 bhp', '270.9 bhp', '265 bhp', '157.75 bhp', '101 bhp',
'186 bhp', '187.4 bhp', '224 bhp', '64.9 bhp', '148 bhp',
'35.5 bhp', '89.75 bhp', '91.72 bhp', '106 bhp', '98.97 bhp',
'66.6 bhp', '86 bhp', '65.3 bhp', '98.82 bhp', '198.25 bhp',
'38 bhp', '142 bhp', '132 bhp', '174.57 bhp', '178 bhp',
'163.2 bhp', '203.2 bhp', '177.5 bhp', '175 bhp', '57 bhp',
'80.84 bhp', '68.4 bhp', '167.67 bhp', '170.63 bhp', '52 bhp',
'149.5 bhp', '48.21 bhp', '201.1 bhp', '100.5 bhp', '144 bhp',
'194.4 bhp', '168.7 bhp', '104.5 bhp', '103.26 bhp', '116.4 bhp',
'98.79 bhp', '272 bhp', '235 bhp', '167.62 bhp', '170.30 bhp',
'139.46 bhp', '158 bhp', '110.5 bhp', '82.5 bhp', '141.1 bhp',
'197.2 bhp', '161 bhp', '194 bhp', '122.4 bhp', '134.10 bhp',
'134 bhp', '203 bhp', '135.1 bhp'], dtype=object)
```

- Notei que esses valores estão com uma unidade chamada de "bhp". Essa unidade pode vir descrita apenas na legenda dessa variável.
- Assim, irei remover essa unidade dos respectivos valores e irei colocar na legenda a unidade "bhp".

In [ ]:
```
# Removendo as unidades de uma lista de valores
# Usando replace() + strip() + list comprehension
import re
```

In [ ]:
```
# Iniciailizando a lista
test_list = carros.potencia_do_motor
```

In [ ]:
```
# Imprimindo a lista original
print("A lista original é : " + str(test_list))
```

```
A lista original é : 0          74 bhp
1       103.52 bhp
2           78 bhp
3           90 bhp
4         88.2 bhp
           ...
```

```
8123     82.85 bhp
8124       110 bhp
8125      73.9 bhp
8126        70 bhp
8127        70 bhp
Name: potencia_do_motor, Length: 7819, dtype: object
```

In [ ]:
```
# Inicializando as unidades
unit = "bhp"
```

In [ ]:
```
# Removendo unidades de uma lista de valores
# Usando replace() + strip() + list comprehension
carros.potencia_do_motor = [sub.replace(unit, "").strip() for sub in test_list]
```

In [ ]:
```
# Imprimindo resultado
print("Lista após remoção das unidades : " + str(carros.potencia_do_motor))
```

```
Lista após remoção das unidades : 0           74
1       103.52
2           78
3           90
4         88.2
          ...
8123     82.85
8124       110
8125      73.9
8126        70
8127        70
Name: potencia_do_motor, Length: 7819, dtype: object
```

In [ ]:
```
carros.dtypes
```

Out[ ]:
```
nome_completo_do_carro          object
ano_de_venda                     int64
preco_de_venda_USD             float64
kilometragem                     int64
tipo_combustivel                object
tipo_do_vendedor                object
transmissao                     object
dono                            object
consumo_do_combustivel_kmpl    float64
motor_CC                       float64
potencia_do_motor               object
torque                          object
assentos                       float64
Marca                           object
dtype: object
```

Tenho de trocar o tipo de valor dessa "potencia_do_motor".

In [ ]:
```
carros["potencia_do_motor"]=carros["potencia_do_motor"].astype(float)

carros.potencia_do_motor.unique()
```

Out[ ]:
```
array([ 74.   , 103.52 ,  78.   ,  90.   ,  88.2  ,  81.86 ,  37.   ,
        67.1  ,  68.1  , 108.45 ,  60.   ,  73.9  ,  67.   ,  82.   ,
        88.5  ,  46.3  ,  88.73 ,  64.1  ,  98.6  ,  88.8  ,  83.81 ,
        83.1  ,  47.3  ,  73.8  ,  34.2  ,  35.   ,  81.83 , 121.3  ,
       138.03 , 160.77 , 117.3  , 116.3  ,  83.14 ,  67.05 , 168.5  ,
       100.   , 120.7  ,  98.63 , 175.56 , 103.25 , 171.5  , 100.6  ,
       174.33 , 187.74 , 170.   ,  78.9  ,  88.76 ,  86.8  , 108.495,
       108.62 ,  93.7  , 103.6  ,  98.59 , 189.   ,  67.04 ,  68.05 ,
        82.85 ,  81.8  ,  73.   , 120.   ,  94.68 , 160.   ,  65.   ,
       155.   ,  69.01 , 126.32 , 138.1  ,  83.8  , 126.2  ,  98.96 ,
```

```
 62.1  ,  86.7  , 188.   , 214.56 , 177.   , 280.   , 148.31 ,
254.79 , 190.   , 177.46 , 204.   , 141.   , 117.6  , 241.4  ,
282.   , 150.   , 147.5  , 108.5  , 103.5  , 183.   , 181.04 ,
157.7  , 164.7  ,  91.1  , 400.   ,  68.   ,  75.   ,  85.8  ,
 87.2  ,  53.   , 118.   , 103.2  ,  83.   ,  84.   , 147.94 ,
 74.02 ,  53.3  ,  80.   ,  88.7  ,  97.7  , 121.36 , 162.   ,
140.   ,  94.   , 100.57 ,  82.9  ,  83.11 ,  70.   , 153.86 ,
121.   , 126.3  ,  73.97 , 171.   ,  69.   ,  99.6  , 102.   ,
105.   ,  63.   ,  79.4  ,  97.9  ,  63.1  ,  66.1  , 110.   ,
174.5  ,  53.26 ,  73.75 ,  67.06 ,  64.08 ,  37.5  , 189.3  ,
158.8  ,  55.2  ,  71.01 ,  73.74 , 147.9  ,  71.   ,  77.   ,
121.4  , 113.4  ,  47.   , 130.   ,  57.6  , 138.   ,  52.8  ,
 53.64 ,  53.5  ,  76.8  ,  82.4  , 113.42 ,  76.   ,  84.8  ,
 56.3  , 218.   , 112.   ,  92.   , 105.5  , 169.   ,  95.   ,
 72.4  , 115.   , 152.   ,  91.2  , 156.   ,  74.9  ,  62.   ,
105.3  ,  73.94 ,  85.   , 118.3  ,  72.   , 147.51 ,  58.   ,
 64.   , 126.24 ,  76.9  , 194.3  ,  99.23 ,  89.84 , 123.7  ,
118.35 ,  99.   , 241.   , 136.   , 261.4  , 104.68 ,  37.48 ,
104.   ,  63.12 ,  91.7  , 102.5  , 177.6  ,  45.   , 123.37 ,
147.8  , 184.   ,  84.48 ,  68.07 ,  74.96 , 167.6  , 152.87 ,
112.2  ,  83.83 , 197.   , 110.4  , 104.55 , 103.   , 103.3  ,
 66.   , 108.6  , 165.   , 163.7  , 116.9  ,  94.93 , 127.   ,
198.5  , 179.5  , 120.69 , 121.31 , 138.08 , 187.7  ,  80.8  ,
 86.79 ,  93.87 , 116.6  , 143.   ,  92.7  ,  88.   ,  78.8  ,
 64.4  , 125.   , 139.01 , 254.8  , 181.   , 258.   , 270.9  ,
265.   , 157.75 , 101.   , 186.   , 187.4  , 224.   ,  64.9  ,
148.   ,  35.5  ,  89.75 ,  91.72 , 106.   ,  98.97 ,  66.6  ,
 86.   ,  65.3  ,  98.82 , 198.25 ,  38.   , 142.   , 132.   ,
174.57 , 178.   , 163.2  , 203.2  , 177.5  , 175.   ,  57.   ,
 80.84 ,  68.4  , 167.67 , 170.63 ,  52.   , 149.5  ,  48.21 ,
201.1  , 100.5  , 144.   , 194.4  , 168.7  , 104.5  , 103.26 ,
116.4  ,  98.79 , 272.   , 235.   , 167.62 , 170.3  , 139.46 ,
158.   , 110.5  ,  82.5  , 141.1  , 197.2  , 161.   , 194.   ,
122.4  , 134.1  , 134.   , 203.   , 135.1  ])
```

In [ ]:
```
carros.dtypes
```

Out[ ]:
```
nome_completo_do_carro          object
ano_de_venda                     int64
preco_de_venda_USD             float64
kilometragem                     int64
tipo_combustivel                object
tipo_do_vendedor                object
transmissao                     object
dono                            object
consumo_do_combustivel_kmpl    float64
motor_CC                       float64
potencia_do_motor              float64
torque                          object
assentos                       float64
Marca                           object
dtype: object
```

Agora sim, essa variável "potencia_do_motor" está devidamente formatada. Falta apenas colocar a unidade no título da variável.

In [ ]:
```
#Trocando o título da variável

carros.rename(columns={"potencia_do_motor": "potencia_do_motor_bhp"}, inplace=True)

carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda_USD	kilometragem	tipo_combustivel	tipo_do_
0	Maruti Swift Dzire VDI	2014	6164.38	145500	Diesel	
1	Skoda Rapid 1.5 TDI	2014	5068.49	120000	Diesel	

	Ambition				
2	Honda City 2017-2020 EXi	2006	2164.38	140000	Petrol
3	Hyundai i20 Sportz Diesel	2010	3082.19	127000	Diesel
4	Maruti Swift VXI BSIII	2007	1780.82	120000	Petrol

Variável 12 - torque

In [ ]:
```
#valores únicos:

carros.torque.unique()
```

Out[ ]:
```
array(['190Nm@ 2000rpm', '250Nm@ 1500-2500rpm', '12.7@ 2,700(kgm@ rpm)',
       '22.4 kgm at 1750-2750rpm', '11.5@ 4,500(kgm@ rpm)',
       '113.75nm@ 4000rpm', '59Nm@ 2500rpm', '170Nm@ 1800-2400rpm',
       '160Nm@ 2000rpm', '248Nm@ 2250rpm', '78Nm@ 4500rpm',
       '84Nm@ 3500rpm', '115Nm@ 3500-3600rpm', '200Nm@ 1750rpm',
       '62Nm@ 3000rpm', '219.7Nm@ 1500-2750rpm', '114Nm@ 3500rpm',
       '115Nm@ 4000rpm', '69Nm@ 3500rpm', '172.5Nm@ 1750rpm',
       '6.1kgm@ 3000rpm', '114.7Nm@ 4000rpm', '90Nm@ 3500rpm',
       '151Nm@ 4850rpm', '104Nm@ 4000rpm', '320Nm@ 1700-2700rpm',
       '250Nm@ 1750-2500rpm', '145Nm@ 4600rpm', '146Nm@ 4800rpm',
       '343Nm@ 1400-3400rpm', '200Nm@ 1400-3400rpm',
       '200Nm@ 1250-4000rpm', '400Nm@ 2000-2500rpm', '138Nm@ 4400rpm',
       '360Nm@ 1200-3400rpm', '200Nm@ 1200-3600rpm',
       '380Nm@ 1750-2500rpm', '173Nm@ 4000rpm', '400Nm@ 1750-3000rpm',
       '400Nm@ 1400-2800rpm', '200Nm@ 1750-3000rpm', '111.7Nm@ 4000rpm',
       '219.6Nm@ 1500-2750rpm', '112Nm@ 4000rpm', '250Nm@ 1500-3000rpm',
       '130Nm@ 4000rpm', '205Nm@ 1750-3250rpm', '280Nm@ 1350-4600rpm',
       '99.04Nm@ 4500rpm', '110Nm@ 3750rpm', '153Nm@ 3800rpm',
       '113.7Nm@ 4000rpm', '114Nm@ 4000rpm', '113Nm@ 4200rpm',
       '101Nm@ 3000rpm', '290Nm@ 1800-2800rpm', '120Nm@ 4250rpm',
       '250Nm@ 1500~4500rpm', '96 Nm at 3000 rpm', '360Nm@ 1750-2800rpm',
       '135Nm@ 2500rpm', '259.8Nm@ 1900-2750rpm', '200Nm@ 1900rpm',
       '259.9Nm@ 1900-2750rpm', '91Nm@ 4250rpm', '96.1Nm@ 3000rpm',
       '109Nm@ 4500rpm', '400nm@ 1750-3000rpm', '202Nm@ 3600-5200rpm',
       '430Nm@ 1750-2500rpm', '347Nm@ 4300rpm', '382nm@ 1750-2250rpm',
       '620Nm@ 1600-2400rpm', '400Nm@ 1750-2500rpm', '250@ 1250-5000rpm',
       '500Nm@ 1600-1800rpm', '250Nm@ 1600-3600rpm', '400Nm',
       '550Nm@ 1750-2750rpm', '490Nm@ 1600rpm', '250 Nm at 2750 rpm',
       '177.5Nm@ 4700rpm', '170Nm@ 1750-4000rpm', '300Nm@ 1200-4000rpm',
       '300Nm@ 1200-1400rpm', '260Nm@ 1500-2750rpm', '213Nm@ 4500rpm',
       '224Nm@ 4000rpm', '640Nm@ 1740rpm', '113Nm@ 4500rpm',
       '95Nm@ 3000-4300rpm', '13.1kgm@ 4600rpm', '205Nm@ 1800-2800rpm',
       '71Nm@ 3500rpm', '190Nm@ 1750-3000rpm', '146Nm at 4800 rpm',
       '14.9 KGM at 3000 RPM', '115Nm@ 3200rpm', '117nm@ 4000rpm',
       '320Nm@ 1500-3000rpm', '72Nm@ 4386rpm', '11.4 kgm at 4,000 rpm',
       '140Nm@ 1500-4000rpm', '134Nm@ 4000rpm', '150Nm@ 4500rpm',
       '340Nm@ 1800-3250rpm', '240Nm@ 1600-2800rpm',
       '330Nm@ 1600-2800rpm', '12.5@ 3,500(kgm@ rpm)', '110Nm@ 4800rpm',
       '111.8Nm@ 4000rpm', '11.8@ 3,200(kgm@ rpm)', '135.4Nm@ 2500rpm',
       '300Nm@ 1750-2500rpm', '190.25nm@ 1750-2250rpm',
       '140Nm@ 1800-3000rpm', '20.4@ 1400-3400(kgm@ rpm)',
       '247Nm@ 1800-2000rpm', '223Nm@ 1600-2200rpm',
       '180 Nm at 1440-1500rpm', '195Nm@ 1400-2200rpm',
       '154.9Nm@ 4200rpm', '114.73Nm@ 4000rpm', '160Nm@ 1500-2750rpm',
       '108Nm@ 4400rpm', '190.24nm@ 1750-2250rpm', '200Nm@ 2000-3500rpm',
       '420Nm@ 1400-2600rpm', '100Nm@ 2700rpm', '51Nm@ 4000rpm',
       '250Nm@ 1250-5300rpm', '132Nm@ 3000rpm', '350Nm@ 1500-2750rpm',
```

'218Nm@ 4200rpm', '14.9@ 3,000(kgm@ rpm)',
'24@ 1,900-2,750(kgm@ rpm)', '13.5@ 2,500(kgm@ rpm)',
'74.5Nm@ 4000rpm', '160Nm@ 1750rpm', '180.4Nm@ 1750-2500rpm',
'230Nm@ 1500-2500rpm', '113.75Nm@ 4000rpm',
'219.66nm@ 1500-2750rpm', '245Nm@ 1750rpm', '360Nm@ 1400-3200rpm',
'320Nm@ 2000rpm', '135 Nm at 2500 rpm ',
'24 KGM at 1900-2750 RPM', '190Nm@ 1750-2250rpm',
'204Nm@ 2000-2750rpm', '14.3@ 1,800-3,000(kgm@ rpm)',
'250nm@ 1500-2750rpm', '125Nm@ 2000rpm', '172Nm@ 4300rpm',
'150Nm@ 1750rpm', '102Nm@ 4000rpm', '85Nm@ 2500rpm',
'8.5@ 2,500(kgm@ rpm)', '180Nm@ 1440-1500rpm', '106.5Nm@ 4400rpm',
'108.5Nm@ 5000rpm', '350Nm@ 1750-2500rpm', '144.15nm@ 4500rpm',
'104Nm@ 4400rpm', '99Nm@ 4500rpm', '200Nm@ 2000rpm',
'280Nm@ 1800-2800rpm', '142.5Nm@ 1750rpm', '140Nm@ 4400rpm',
'115@ 2,500(kgm@ rpm)', '196Nm@ 5000rpm',
'260 Nm at 1800-2200 rpm', '9.8@ 3,000(kgm@ rpm)',
'209Nm@ 2000rpm', '135 Nm at 2500 rpm', '140Nm@ 4200rpm',
'220Nm at 1400-2600 rpm', '48Nm@ 3000rpm', '171Nm@ 1800rpm',
'277.5Nm@ 1700-2200rpm', '215Nm@ 3600rpm', '219.6Nm@ 1750-2750rpm',
'195Nm@ 1440-2200rpm', '13@ 2,500(kgm@ rpm)', '180Nm@ 2000rpm',
'200Nm@ 1400-2200rpm', '380Nm(38.7kgm)@ 2500rpm', '110Nm@ 4400rpm',
'72Nm@ 4388rpm', '263.7Nm@ 2500rpm', '320Nm@ 1600-2800rpm',
'25.5@ 1,500-3,000(kgm@ rpm)', '16.3@ 2,000(kgm@ rpm)',
'190 Nm at 1750 rpm ', '94.14Nm@ 3500rpm', '12@ 3,500(kgm@ rpm)',
'113Nm@ 5000rpm', '280Nm@ 2400-2800rpm', '96Nm@ 3500rpm',
'16@ 2,000(kgm@ rpm)', '320Nm@ 1750-3000rpm', '114.73nm@ 4000rpm',
'320Nm@ 1750-2500rpm', '138nm@ 4400rpm', '190Nm@ 1750rpm',
'789Nm@ 2250rpm', '259.87Nm@ 1900-2750rpm', '205Nm@ 1750rpm',
'436.39Nm@ 1800-2500rpm', '182.5Nm@ 1500-1800rpm',
'90.3Nm@ 4200rpm', '12.5@ 2,500(kgm@ rpm)', '215Nm@ 1750-3000rpm',
'215Nm@ 1750-3000', '305Nm@ 2000rpm', '540Nm@ 2000rpm',
'327Nm@ 2600rpm', '300Nm@ 1600-3000rpm', '620Nm@ 2000-2500rpm',
'450Nm@ 1600-2400rpm', '19@ 1,800(kgm@ rpm)',
'9.2@ 4,200(kgm@ rpm)', '145@ 4,100(kgm@ rpm)',
'51Nm@ 4000+/-500rpm', '110Nm@ 3000rpm', '148Nm@ 3500rpm',
'116Nm@ 4750rpm', '48@ 3,000+/-500(NM@ rpm)', '148Nm@ 4000rpm',
'222Nm@ 4300rpm', '135.3Nm@ 5000rpm', '98Nm@ 1600-3000rpm',
'170Nm@ 1400-4500rpm', '343Nm@ 1400-2800rpm',
'402Nm@ 1600-3000rpm', '113Nm@ 3300rpm', '99.07Nm@ 4500rpm',
'210nm@ 1600-2200rpm', '190 Nm at 1750 rpm ', '32.1kgm@ 2000rpm',
'224nm@ 1500-2750rpm', '400nm@ 1750-2500rpm',
'215Nm@ 1750-2500rpm', '25@ 1,800-2,800(kgm@ rpm)',
'197Nm@ 1750rpm', '136.3Nm@ 4200rpm', '470Nm@ 1750-2500rpm',
'11@ 3,000(kgm@ rpm)', '142Nm@ 4000rpm', '145Nm@ 4100rpm',
'320Nm@ 1500-2800rpm', '123Nm@ 1000-2500rpm',
'218Nm@ 1400-2600rpm', '510@ 1600-2400', '220Nm@ 1500-2750rpm',
'380Nm@ 2000rpm', '104Nm@ 3100rpm', '292Nm@ 2000rpm',
'20@ 3,750(kgm@ rpm)', '46.5@ 1,400-2,800(kgm@ rpm)',
'380Nm@ 2500rpm', '15@ 3,800(kgm@ rpm)', '136Nm@ 4250rpm',
'228Nm@ 4400rpm', '149Nm@ 4500rpm', '187Nm@ 2500rpm',
'146Nm@ 3400rpm', '8.6@ 3,500(kgm@ rpm)', '219.7Nm@ 1750-2750rpm',
'190Nm@ 2000-3000', '450Nm@ 2000rpm', '300Nm@ 2000rpm',
'230Nm@ 1800-2000rpm', '42@ 2,000(kgm@ rpm)',
'110Nm@ 3000-4300rpm', '110(11.2)@ 4800', '330Nm@ 1800rpm',
'225Nm@ 1500-2500rpm', '380Nm@ 1750-2750rpm',
'28.3@ 1,700-2,200(kgm@ rpm)', '259.88Nm@ 1900-2750rpm',
'580Nm@ 1400-3250rpm', '400 Nm /2000 rpm', '127Nm@ 3500rpm',
'300Nm@ 1500-2500rpm', '132.3Nm@ 4000rpm', '113nm@ 4400rpm',
'151NM@ 4850rpm', '153Nm@ 3750-3800rpm', '10.7@ 2,500(kgm@ rpm)',
'124.6Nm@ 3500rpm', '219.9Nm@ 1750-2750rpm',
'420.7Nm@ 1800-2500rpm', '130Nm@ 3000rpm', '424Nm@ 2000rpm',
'130@ 2500(kgm@ rpm)', '99.8Nm@ 2700rpm', '113Nm@ 4,500rpm',
'11.2@ 4,400(kgm@ rpm)', '240Nm@ 1850rpm', '16.1@ 4,200(kgm@ rpm)',
'320Nm@ 1750-2700rpm', '115Nm@ 4500rpm', '245Nm@ 4000rpm',
'321Nm@ 1600-2400rpm', '619Nm@ 1600-2400rpm',
'380Nm@ 1750-3000rpm', '560Nm@ 1500rpm', '230Nm@ 1500-2250rpm',
'260Nm@ 1800-2200rpm', '600Nm@ 2000rpm', '259.87nm@ 1500-3000rpm',
'16.6@ 4,500(kgm@ rpm)', '219.66NM@ 1500-2750rpm',
'12.5@ 3,000(kgm@ rpm)', '620Nm@ 1500-2500rpm',

```
'250Nm@ 1500-4500rpm', '14.9@ 3,400(kgm@ rpm)',
'25.5@ 1,900(kgm@ rpm)', '33.7@ 1,800(kgm@ rpm)',
'285Nm@ 2400-4000rpm', '10.7@ 2,600(kgm@ rpm)',
'250Nm@ 1000-2000rpm', '240Nm@ 1750rpm', '226Nm@ 4400rpm',
'510Nm@ 1600-2800rpm', '259.87NM@ 1500-3000rpm',
'155 Nm at 1600-2800 rpm', '240Nm@ 2000rpm', '103Nm@ 4500rpm',
'13.5@ 4,800(kgm@ rpm)', '400Nm@ 1750-2750rpm',
'175Nm@ 1500-4100rpm', '72.9Nm@ 2250rpm', '135.4Nm@ 2500',
'245Nm@ 5000rpm', '96Nm@ 2500rpm', '215nm@ 1750-2500rpm',
'10.4@ 3,200(kgm@ rpm)', '128Nm@ 3100rpm', '102Nm@ 2600rpm',
'131Nm@ 4400rpm', '11.4@ 4,000(kgm@ rpm)', '250Nm@ 4250rpm',
'343Nm@ 1600-2800rpm', '185Nm@ 1750-2750rpm', '12@ 2500(kgm@ rpm)',
'12.4@ 2,600(kgm@ rpm)', '170Nm@ 4200rpm', '176Nm@ 1500rpm',
'380NM@ 1800-2800rpm', '250Nm@ 1600-2000rpm',
'24.5@ 3,500-4,500(kgm@ rpm)', '22.9@ 1,950-4,700(kgm@ rpm)',
'113Nm@ 4400rpm', '121Nm@ 2800rpm', '210 / 1900',
'250Nm@ 1250-5000rpm', '400Nm@ 175-2750rpm', '350Nm@ 1500-3500rpm',
'175nm@ 1750-4000rpm', '115@ 2500(kgm@ rpm)', '110Nm@ 4500rpm',
'190Nm@ 2000-3000rpm', '106Nm@ 2200rpm',
'21.4@ 1,750-4,600(kgm@ rpm)', '96Nm@ 3000rpm',
'23.6@ 4,250(kgm@ rpm)', '11.3kgm@ 4700rpm', '450Nm@ 1750-2500rpm',
'35.7@ 1,750-3,000(kgm@ rpm)', '6@ 2,500(kgm@ rpm)',
'13.9 kgm at 4200 rpm', '320Nm@ 1400-4100rpm',
'150Nm@ 1700-4500rpm', '113.8Nm@ 4000rpm', '110@ 3,000(kgm@ rpm)',
'151Nm@ 2400rpm', '62Nm@ 2500rpm', '18@ 1,600-2,200(kgm@ rpm)',
'20@ 4,700(kgm@ rpm)', '300Nm@ 1600-4000rpm',
'171.6Nm@ 1500-4000rpm', '21.4@ 1,900(kgm@ rpm)',
'190@ 21,800(kgm@ rpm)', '250 Nm at 1,500-3,000 rpm',
'340nm@ 1750-3000rpm', '36.6@ 1,750-2,500(kgm@ rpm)',
'12.5kgm@ 3500rpm', '6.1@ 3,000(kgm@ rpm)', '350nm@ 1800-2600rpm',
'175nm@ 1500-4100rpm', '4.8kgm@ 3000rpm', '355Nm@ 4500rpm',
'51@ 1,750-3,000(kgm@ rpm)', '119Nm@ 4250rpm',
'410Nm@ 1600-2800rpm', '174Nm@ 4300rpm', '385Nm@ 1600-2500rpm',
'180 Nm at 2000rpm', '190 Nm at 1750 rpm',
'53@ 2,000-2,750(kgm@ rpm)', '360Nm@ 1400-2600rpm',
'420Nm@ 2000rpm', '124Nm@ 3500rpm', '17.5@ 4,300(kgm@ rpm)',
'360Nm@ 2000rpm', '145Nm@ 3750rpm', '190Nm@ 4200rpm',
'190 Nm at 2000rpm', '13.5@ 2500(kgm@ rpm)', '250nm@ 1500-3000rpm',
'159.8Nm@ 1500-2750rpm', '500Nm@ 2000rpm', '333Nm@ 1600-3200rpm',
'400nm@ 2800rpm', '33@ 2,000-2,680(kgm@ rpm)',
'10.2@ 2,600(kgm@ rpm)', '480Nm', '190Nm@ 4300rpm',
'320Nm@ 1800-2800rpm', '380Nm@ 1750rpm', '250.06nm@ 1500-2750rpm',
'90nm@ 3500rpm', '190Nm@ 3700rpm', '436.4Nm@ 1800-2500rpm',
'96  Nm at 3000  rpm '], dtype=object)
```

Essa variável torque possui unidades não formatadas e com distintas amplitudes de valores dentro de uma mesma célula. Dessa forma, devido a falta de padronização dos dados optarei aqui por não analisá-las na sequência desse projeto, a fim de não influenciar negativamente nas análises dos dados.

- Dessa forma, irei remover essa coluna do conjunto de dados.

In [ ]:
```
del carros["torque"]
```

In [ ]:
```
carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda_USD	kilometragem	tipo_combustivel	tipo_do_
0	Maruti Swift Dzire VDI	2014	6164.38	145500	Diesel	
1	Skoda Rapid 1.5 TDI Ambition	2014	5068.49	120000	Diesel	
2	Honda City 2017-2020 EXi	2006	2164.38	140000	Petrol	
3	Hyundai i20 Sportz Diesel	2010	3082.19	127000	Diesel	

4	Maruti Swift VXI BSIII	2007	1780.82	120000	Petrol

Variável 13 - "assentos"

In [ ]:
```
#valores únicos:

carros.assentos.unique()
```

Out[ ]:
```
array([ 5.,  4.,  7.,  8.,  6.,  9., 10., 14.,  2.])
```

In [ ]:
```
carros.dtypes
```

Out[ ]:
```
nome_completo_do_carro          object
ano_de_venda                     int64
preco_de_venda_USD             float64
kilometragem                     int64
tipo_combustivel                object
tipo_do_vendedor                object
transmissao                     object
dono                            object
consumo_do_combustivel_kmpl    float64
motor_CC                       float64
potencia_do_motor_bhp          float64
assentos                       float64
Marca                           object
dtype: object
```

In [ ]:
```
carros.head(5)
```

Out[ ]:

	nome_completo_do_carro	ano_de_venda	preco_de_venda_USD	kilometragem	tipo_combustivel	tipo_do_
0	Maruti Swift Dzire VDI	2014	6164.38	145500	Diesel	
1	Skoda Rapid 1.5 TDI Ambition	2014	5068.49	120000	Diesel	
2	Honda City 2017-2020 EXi	2006	2164.38	140000	Petrol	
3	Hyundai i20 Sportz Diesel	2010	3082.19	127000	Diesel	
4	Maruti Swift VXI BSIII	2007	1780.82	120000	Petrol	

A etapa acima finaliza a "Parte A do Projeto 01"

Agora vou salvar essa planilha final para poder dar prosseguimento as análises no próximo projeto.

In [ ]:
```
from google.colab import files
carros.to_csv('carros_formatado.csv', index=False)
files.download('carros_formatado.csv')
```

Texto por:

Lucas Andrei Campos-Siva

Cientista de Dados / Business Intelligence / Analista de Dados

LinkedIn: https://www.linkedin.com/in/lucas-andrei-campos-silva/

E-mail: andrei.10@hotmail.com

Portifólio de projetos em Data Science: https://github.com/Campos-Silva